เขียนโปรแกรมกับ 8087/80287

สุรศักดิ์ สงวนพงษ์

8087 เป็นโปรเซสเซอร์ร่วมที่ช่วยงานด้านคำนวณของซีพียูตระกูล 8086/8088 โดยตรง อุปมาดั่งโรบินเป็นคู่หูกับมนุษย์ค้างคาวฉันใดก็ฉันนั้น ควรที่เราจะทำความรู้จักและนำมาใช้งาน

ในสายการผลิตไมโครโปรเซสเซอร์ตั้งแต่ระดับ 16 บิตของอินเทลประกอบไปด้วย 8086/8088, 80186/80188, 80286 และ 80386 ไมโครโปรเซสเซอร์เหล่านี้ทำหน้าที่เป็นซีพียู และสามารถนำโปรเซสเซอร์ร่วม (coprocessor) เข้ามาช่วยการทำงานเพื่อขยายขีดความสามารถบางอย่างของซีพียูได้ โปรเซสเซอร์ร่วมเหล่านี้อาจจะเป็นไอโอโปรเซสเซอร์ช่วยงานทางด้านอินพุตเอาต์พุต เช่น 8089, 82258 เป็นดีเอ็มเอคอนโทรลเลอร์ 82586 เป็นโลคอลเอเรียเน็ตเวอร์กโปรเซสเซอร์ หรือ 82730 ซึ่งใช้เป็นโปรเซสเซอร์จัดการด้านการแสดงผล ส่วนด้านแมตช์โปรเซสเซอร์นั้นเราค่อนข้างจะคุ้นชื่อมากกว่า ได้แก่ 8087, 80287 และ 80387 ซึ่งจะกล่าวถึงในที่นี้ แผนภาพในรูปที่ 1 จะแสดงให้เห็นถึงโปรเซสเซอร์ร่วมที่อินเทลออกแบบมาให้ใช้กับซีพียูเบอร์ต่าง ๆ

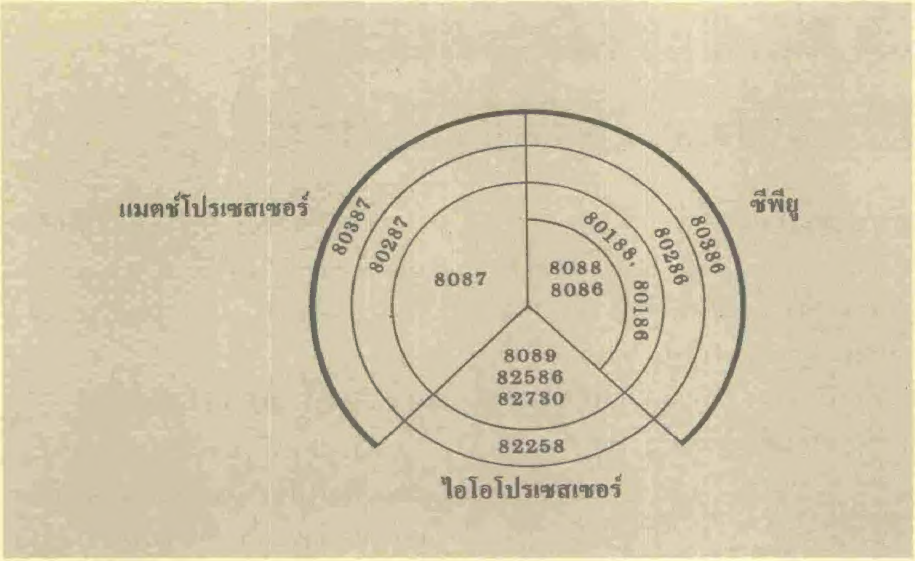
ตระกูลแมตช์โปรเซสเซอร์

อินเทลได้พัฒนาแมตช์โปรเซสเซอร์ควบคู่มากับไมโครโปรเซสเซอร์ 16 บิต ของอินเทล แมตช์โปรเซสเซอร์ตัวแรกคือ 8087 ออกแบบมาใช้กับ 8086/8088 และยังใช้ร่วมกับ 80186/80188 ได้อีกด้วย อินเทลยังได้พัฒนา 80287 ต่อจาก 8087 เพื่อนำมาใช้กับซีพียู 80286 และแมตช์โปรเซสเซอร์ 80387 ตัวใหม่ล่าสุดซึ่งออกแบบมาใช้เฉพาะกับ 80386 ซีพียูล่าสุดของอินเทลเช่นกัน

8087 เป็นไมโครโปรเซสเซอร์ขนาด 80 บิต สามารถทำคำสั่งคณิตศาสตร์ง่ายๆตั้งแต่บวก ลบ คูณ หาร เปรียบเทียบหรือการหารากที่สองด้วยคำสั่งเพียงคำสั่งเดียว เร็วกว่าซอฟต์แวร์รูทีนที่เขียนด้วย 8088 หรือ 8086 ถึงประมาณ 100 เท่า นอกจากนี้ยังสามารถโปรแกรม 8087 ให้คำนวณฟังก์ชันทรานเซนเดนทัลและตรีโกณมิติได้เร็วกว่าซอฟต์แวร์รูทีนทั่วไปประมาณ 50 เท่า

8087 และ 80287 นั้น มีสมรรถนะการทำงานใกล้เคียงกัน ทั้งในด้านชุดคำสั่งก็ไม่ได้แตกต่างกัน (ในบทความนี้จะกล่าวถึง 8087 และ 80287 ไปพร้อมกัน เมื่อกล่าวถึง 8087 ก็จะรวมไปถึง 80287 ด้วย) อย่างไรก็ตาม 80287 ได้เพิ่มลักษณะพิเศษในการเชื่อมต่อกับ 80286 ในกรณีที่ทั้งสองโปรเซสเซอร์ทำงานที่ความถี่ต่างกัน สำหรับ 80387 นั้น ทำงานได้เช่นเดียวกับ 8087/80287 รวมทั้งยังได้เพิ่มฟังก์ชันทางตรีโกณมิติ เช่น ฟังก์ชันไซน์และโคไซน์อีกด้วย ในตารางที่ 1 นั้น เป็นการเปรียบเทียบโปรเซสเซอร์คณิตศาสตร์ทั้ง 3 ตัวนี้

ในบทความนี้ จะได้กล่าวถึงทางด้านฮาร์ดแวร์และซอฟต์แวร์ของ 8087 โดยแบ่งออกเป็น 3 ตอน ในตอนแรกจะกล่าวถึงฮาร์ดแวร์และสถาปัตยกรรม ตอนที่สองเป็นเรื่องของชุดคำสั่งและการควบคุม 8087 ส่วนตอนสุดท้ายจะกล่าวถึงซอฟต์แวร์และการเขียนโปรแกรมรวมทั้งข้อแนะนำบางอย่างสำหรับผู้ที่ต้องการจะซื้อ 8087 ไว้ใช้



รูปที่ 1 โปรเซสเซอร์ร่วมที่ใช้งานกับซีพียูเบอร์ต่าง ๆ

ตารางที่ 1 คุณลักษณะของ 8087/80287/80387

โปรเซสเซอร์	คุณสมบัติ	ซีพียูที่ใช้ร่วม	ความเร็ว	ชนิดตัวถังและจำนวนขา
8087	ทำหน้าที่คำนวณค่าทางคณิตศาสตร์ลอจิกและฟังก์ชันทรานเซนเดนทัลกับข้อมูลจำนวนจริงขนาด 32, 64 บิต และข้อมูลบีซีดีขนาด 18 บิต รูปแบบของข้อมูลจำนวนจริงเป็นไปตามมาตรฐาน IEEE	8086/8088 80186/80188	*ที่ความถี่ 5 MHz* คูณ 32 บิต : 19 uS คูณ 64 บิต : 27 uS หาร : 39 uS *ที่ความถี่ 8 MHz* คูณ 32 บิต : 11.9 uS คูณ 64 บิต : 16.9 uS	เซรามิก 40 ขา
80287	เช่นเดียวกับ 8087 และมีการเชื่อมต่อกับ 80286 ในระบบอะซิงโครนัสเพื่อให้ทั้งสองโปรเซสเซอร์	80286 80386	เช่นเดียวกับ 8087	เซรามิก 40 ขา

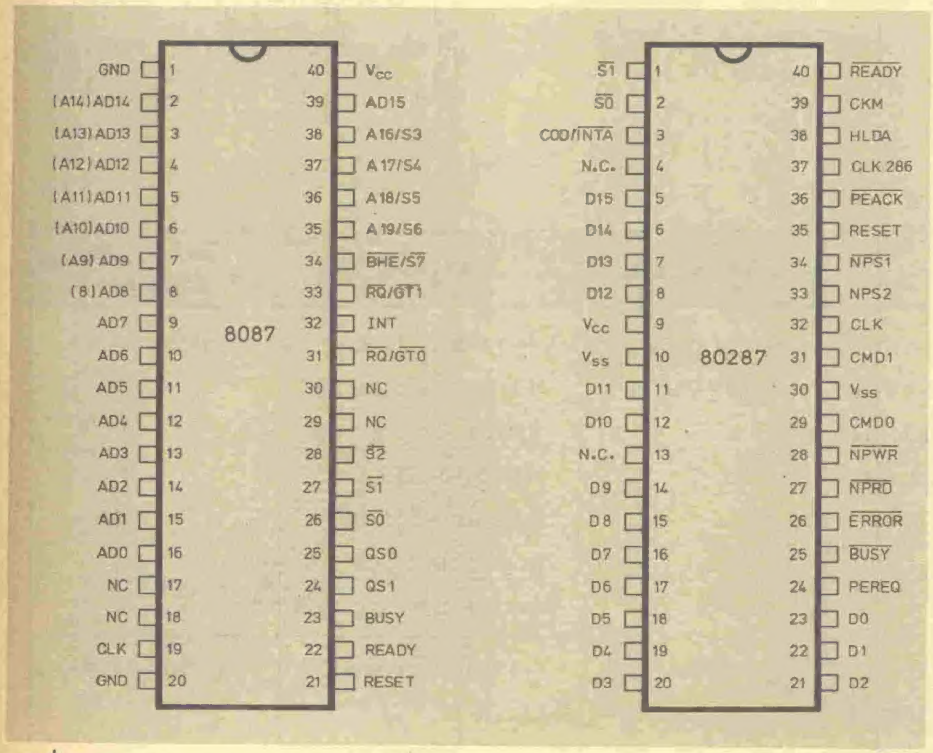
ตารางที่ 1 (ต่อ)

โปรเซสเซอร์	คุณสมบัติ	ซีพียูที่ใช้ร่วม	ความเร็ว	ชนิดตัวถังและจำนวนขา
80287 (ต่อ)	ทำงานที่ความถี่สัญญาณนาฬิกาต่างกันได้			
80387	เช่นเดียวกับ 80287 และได้เพิ่มเติมฟังก์ชันคำนวณเช่น ไซน์และโคไซน์ มีสมรรถนะดีกว่า 80287 โดยรวมประมาณ 4-6 เท่า (ที่ความถี่ 5 MHz)	80386	12.5 MHz และ 16 MHz	พีจีเอ 68 ขา

ฮาร์ดแวร์และสถาปัตยกรรม

ดูลักษณะภายนอก

ทั้ง 8087 และ 80287 เป็นชิพขนาด 40 ขา ทั้งสองตัวมีลักษณะการจัดวางขาแตกต่างกันเพื่อให้เหมาะสมกับซีพียูที่ใช้งานร่วมกันได้ ส่วน 80387 นั้น เนื่องจากมีจำนวนขาใช้งานสูงถึง 68 ขา โครงสร้างภายนอกจึงออกแบบเป็นพินกริดอะเรย์ ในรูปที่ 2 จะแสดงถึงการจัดวางขาของ 8087 และ 80287



รูปที่ 2 การจัดขาของ 8087 และ 80287

มองสถาปัตยกรรมภายใน

การศึกษาถึงการทำงานของโปรเซสเซอร์ใดๆให้เข้าใจ เราจำเป็นต้องมองถึงสถาปัตยกรรมภายในว่ามีโครงสร้างอย่างไร ถ้าเรามองถึงไดอะแกรมในรูปที่ 3 จะเห็นถึงโครงสร้างหลัก การเชื่อมต่อภายในและภายนอก เราอาจแบ่งหน่วยหลักของการทำงานของ 8087 ออกได้เป็น 2 ส่วนคือ

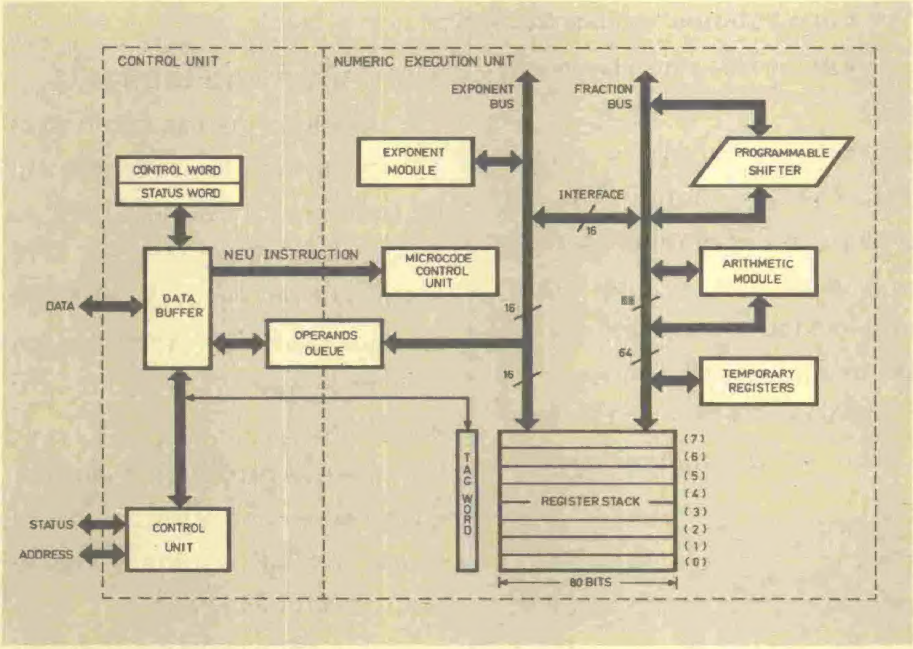
1. หน่วยบัสอินเตอร์เฟส (Bus Interface Unit)

2. หน่วยปฏิบัติงานเชิงเลข (Numeric Execution Unit)

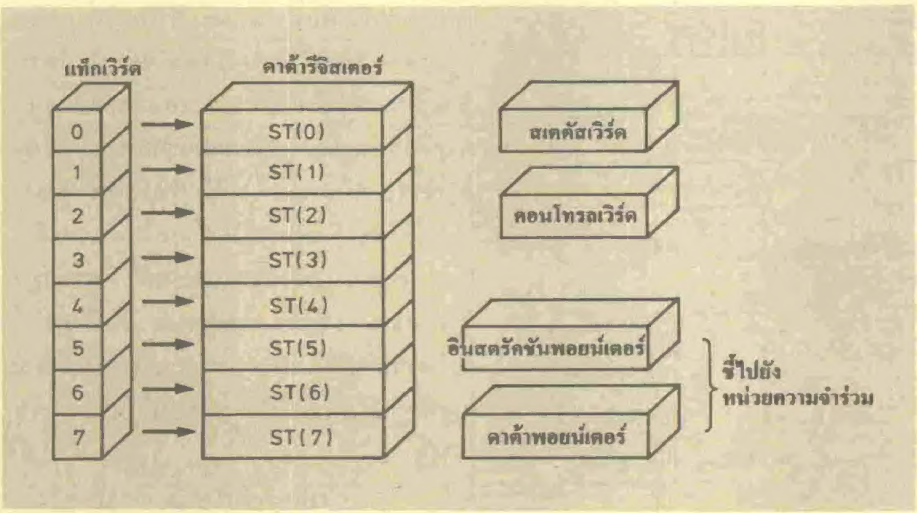
หน่วยบัสอินเตอร์เฟสนี้ มีหน้าที่ประสานการทำงานให้เข้ากับซีพียู เมื่อซีพียูเฟตช์คำสั่งมาจากหน่วยความจำ 8087 จะรับรู้สถานะการทำงานของซีพียูอยู่เสมอโดยมองจาก Status ผ่านหน่วยควบคุม (control unit) นั่นคือ 8087 ทราบอยู่ตลอดเวลาว่ามีการเฟตช์คำสั่งเกิดขึ้นเมื่อใด

8087 จะเริ่มปฏิบัติงานตามชุดคำสั่งพิเศษที่รู้จักเท่านั้น คำสั่งนี้จะมีรหัสพิเศษคือ ESC นำหน้า แล้วจึงตามด้วยคำสั่งคณิตศาสตร์ ทั้งซีพียูและ 8087 สามารถถอดรหัสและเอ็กซีคิวต์คำสั่งนี้ได้ทั้งคู่ ถ้าคำสั่งทางคณิตศาสตร์นั้นต้องใช้ค่าในหน่วยความจำขึ้นเมื่อใด 8087 จะอาศัยการทำงานของซีพียูโดยผลักภาระให้ซีพียูคำนวณแอดเดรสในหน่วยความจำ เมื่อซีพียูคำนวณเสร็จแอดเดรสของหน่วยความจำก็จะปรากฏบนแอดเดรสบัสแล้ว 8087 ก็จะจับเอาแอดเดรสบัสมาใช้งานต่อไป

สำหรับหน่วยปฏิบัติงานเชิงเลขก็จะรับหน้าที่ทำงานตามคำสั่งทุกคำสั่ง ภายในหน่วยนี้จะประกอบด้วยฮาร์ดแวร์คูณเลขความเร็วสูง หน่วยคำนวณฟังก์ชันคณิตศาสตร์ เช่น คำนวณเลขเอ็กซโปเนนเชียล เป็นต้น



รูปที่ 3 โครงสร้างภายในของ 8087



รูปที่ 4 รีจิสเตอร์ภายในของ 8087

รีจิสเตอร์ภายใน

การเขียนโปรแกรมจำเป็นต้องทราบถึงรีจิสเตอร์ต่างๆที่มีให้ใช้ 8087 มีรีจิสเตอร์ดังในรูปที่ 4 มีรีจิสเตอร์หลักสำคัญคือ ดาต้ารีจิสเตอร์ขนาด 80 บิต จำนวน 8 ตัว นอกจากนี้ยังมีคอนโทรลเวิร์ด สเตตัสเวิร์ด แท็กเวิร์ด อินสตรัคชันพอยน์เตอร์ และดาต้าพอยน์เตอร์ รีจิสเตอร์เหล่านี้มีที่ใช้งานดังต่อไปนี้

● **ดาต้ารีจิสเตอร์ (Data Register)**

ดาต้ารีจิสเตอร์ (หรืออาจเรียกว่ารีจิสเตอร์สแต็ก) เป็นรีจิสเตอร์หลักในการเก็บข้อมูล แต่มีวิธีการใช้แตกต่างกับรีจิสเตอร์ของโปรเซสเซอร์ทั่วไปที่เรารู้จัก เนื่องจากมีโครงสร้างเป็นแบบสแต็ก คือ ใช้การ push และ pop เป็นหลัก และเป็นสแต็กที่มีขนาดจำกัดคือมีเพียงจำนวน 8 ตัวเท่านั้น แต่ละรีจิสเตอร์จะมีขนาดถึง 80 บิต เราเรียกตัวเลขที่จัดเก็บไว้ในรีจิสเตอร์ขนาด 80 บิตนี้ว่า จำนวนจริงชั่วคราว (Temporary Real)

● **แท็กเวิร์ด (Tag Word)**

ประกอบด้วยรีจิสเตอร์ย่อย 8 ตัว ๆ ละ 2 บิต ซึ่งบอกถึงสถานะการใช้งานของดาต้ารีจิสเตอร์แต่ละตัวว่าใช้งานอยู่หรือว่าง แท็กเวิร์ดจะเป็นรีจิสเตอร์ที่ 8087 ใช้เฉพาะภายใน

● **คอนโทรลเวิร์ด (Control Word)**

ใช้เป็นตัวกำหนดสภาวะการทำงานของ 8087

● **สเตตัสเวิร์ด (Status Word)**

ใช้เป็นตัวบอกสถานะการทำงานที่เกิดขึ้น เพื่อนำมาตรวจสอบและควบคุมการทำงานของโปรแกรม

● **อินสตรัคชัน และดาต้าพอยน์เตอร์ (Instruction and Data Pointer)**

ใช้เป็นที่เก็บแอดเดรสของคำสั่งและข้อมูลที่ทั้ง 8087 และซีพียูใช้ร่วมกัน

ชุดคำสั่งและการควบคุมการทำงาน

ในส่วนนี้จะได้กล่าวถึงส่วนประกอบทางซอฟต์แวร์ซึ่งได้แก่ ชุดคำสั่งของ 8087 แบบของข้อมูล รวมทั้งวิธีควบคุมการทำงาน

แบบข้อมูล

ในตอนแรกเราได้กล่าวถึงรีจิสเตอร์ของ 8087 ไปแล้ว และยังคงจำได้ว่าดาต้ารีจิสเตอร์ 8087 มีขนาด 80 บิต ใช้เก็บเลขจำนวนจริงที่เรียกว่าจำนวนจริงชั่วคราว อย่างไรก็ตาม 8087 ไม่ได้รู้จักแต่เพียงเลขจำนวนเพียงอย่างเดียว 8087 สามารถจัดการกับข้อมูลได้ถึง 7 แบบ ได้แก่

จำนวนเต็มแบบเวิร์ด (Word Integer)

จำนวนเต็มแบบสั้น (Short Integer)

จำนวนเต็มแบบยาว (Long Integer)

แพคบีซีดี (Packed BCD)

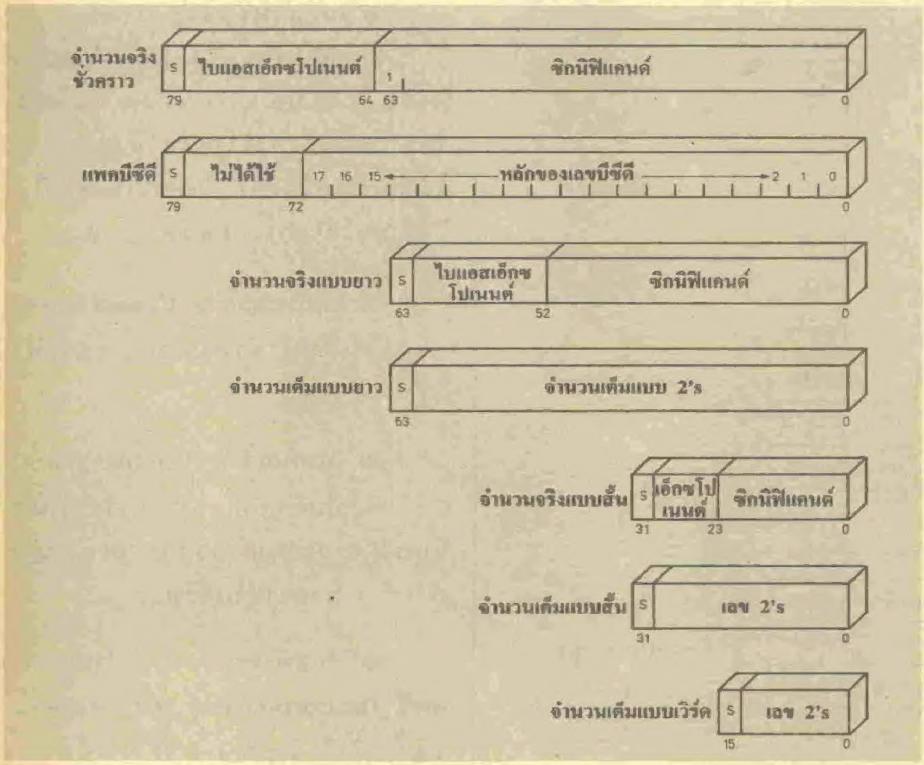
จำนวนจริงแบบสั้น (Short Real)

จำนวนจริงแบบยาว (Long Real)

จำนวนจริงชั่วคราว (Temporary Real)

จะสังเกตเห็นได้ว่า 8087 รู้จักเลขจำนวนเต็ม 4 แบบคือ เวิร์ด, สั้น, ยาว และแพคบีซีดี ส่วนเลขจำนวนจริงก็มี 3 แบบคือ สั้น, ยาว และชั่วคราว ขนาดของตัวเลขแต่ละชนิดก็มีจำนวนบิตไม่เท่ากันตามรูปที่ 1

ที่น่าสังเกตก็คือดาต้ารีจิสเตอร์มีขนาด 80 บิต แต่ข้อมูลแบบต่างๆ มีขนาดต่างกันตั้งแต่ 16 ถึง 80 บิต ปัญหาที่น่าสงสัยก็คือ 8087 จัดการกับเลขเหล่านี้ได้อย่างไร คำตอบก็คือในกรณีที่ข้อมูลซึ่งอาจเป็นจำนวนเต็มหรือจำนวนจริงที่มีขนาดไม่ถึง 80 บิต 8087 จะแปลงข้อมูลนั้นให้มีความกว้างกลายเป็น 80 บิต แล้วจึงโหลดเข้าสู่รีจิสเตอร์ 80 บิต โดยยังคงรักษาค่าของข้อมูลไว้อยู่เพียงแต่ลักษณะหน้าตาข้อมูลจะอยู่ในรูปของตัวเลขขนาด 80 บิตเท่านั้น



รูปที่ 1 แบบข้อมูลของ 8087

8087 เก็บตัวเลขได้อย่างไร

ขอให้เราหันมาทำความเข้าใจกับวิธีการเก็บตัวเลขลงบนรีจิสเตอร์ขนาด 80 บิตกันก่อน เราจะสังเกตได้ว่าเลข 80 บิต ถูกแบ่งออกเป็น 3 ส่วน คือ S ใช้เก็บเครื่องหมาย ไบแอสเอ็กซโปเนนต์ (Biased Exponent) เก็บเลขยกกำลัง และซิกนิฟิแคนด์ (Significand) เก็บข้อมูลตัวเลข

บิตเครื่องหมายหรือ S บิตที่อยู่ซ้ายสุดใช้บอกเครื่องหมายบวกลบของตัวเลข ถ้าบิตนี้เป็น 0 หมายถึง เลขบวก ถ้าเป็น 1 หมายถึง เลขลบ

ให้เราหันมาดูเฉพาะส่วนซิกนิฟิแคนด์โดยละส่วนเอ็กซโปเนนต์ไว้ก่อน บิตที่ 0 ถึง 63 จำนวน 64 บิต ที่ใช้เป็นซิกนิฟิแคนด์นี้จะเป็นตัวเก็บหลักต่าง ๆ ของตัวเลขนั่นเอง อย่างไรก็ตาม 8087 มีวิธีการเก็บตัวเลขที่เราอาจจะเห็นว่าแปลก (แต่ความจริงแล้วเป็นเรื่องปกติ) ก็คือ เลขที่จะนำเก็บจะถูก "นอร์มาไลซ์" เพื่อให้บิตซ้ายสุดนี้กลายเป็น 1 เสียก่อน ดังนั้น เลขทุกตัวที่จะนำมาเก็บจะถูกนอร์มาไลซ์หมด บิตที่ 63 นี้ก็จะมีค่าเป็น 1 เสมอ จะยกเว้นอยู่กรณีเดียวที่เลขนั้นเป็น 0 บิตนี้ถึงจะกลายเป็น 0

เพื่อให้เราเข้าใจวิธีการนี้ขอให้ดูตัวอย่างในรูปที่ 2 ซึ่งแสดงถึงเลข 10

เมื่อถูกเก็บในแบบต่าง ๆ

เราเห็นได้ชัดเจนว่าเวลา 8087 นอร์มาลไลซ์เลข ก็เท่ากับเป็นการเลื่อนเลขนั้นให้บิตที่มีค่าเป็น 1 ไปอยู่ที่บิต 63 นั่นเอง แล้วตัว 8087 ก็จะมองว่าตัวเลขหลังบิต 63 จะมีจุดทศนิยมคั่นอยู่เสมอ นั่นคือ 8087 จะเห็นว่าเลขนี้คือ 1.0100...00 มาถึงตรงนี้อาจทำให้เราสงสัยว่าถ้าเป็นเช่นนี้จะไม่ทำให้ค่าตัวเลขเปลี่ยนไปหรือ คำตอบก็คือเปลี่ยน (เฉพาะในส่วนซิกนิฟิแคนด์) แต่ผลของเอ็กซโปเนนต์ที่เรายังไม่ได้กล่าวถึงจะเป็นตัวไขปัญหาให้เรา

ในขณะนี้ตัวเลข 10 (ฐานสิบ) ได้ถูกเปลี่ยนเป็น 1.010 (ฐานสอง) และนำเข้าไปเก็บในซิกนิฟิแคนด์แล้ว ถ้าเราลองแปลงเลข 1.01 ให้อยู่ในรูปของฐานสิบก็คือ

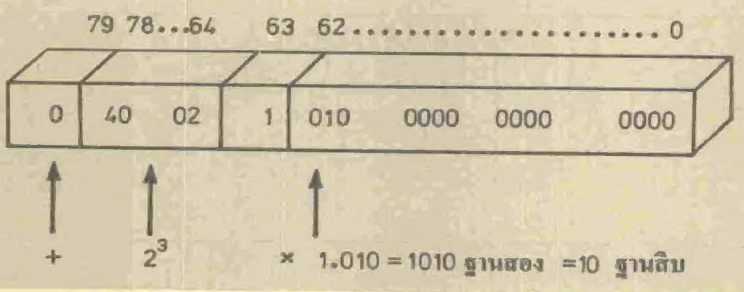
$$1.010_2 = (1\times2^0) + (0\times2^{-1}) + (1\times2^{-2})$$
$$= 1.25$$

เราจะพยายามหาเลขตัวหนึ่งซึ่งจะมาคูณกับ 1.25 แล้วให้ได้ค่า 10 ตามเดิม ตัวเลขนี้ก็คือ 8 หรือ 2^3 นั่นเอง และเราจะพบว่าตัวเลขที่เราหามานี้จะเป็นตัวยกกำลังของเลข 2 เสมอ ในที่นี้คือเลข 3 ตัวเลข 3 นี้เองที่จะถูกนำไปเก็บไว้ในเอ็กซโปเนนต์ฟิลด์ ดังนั้น วิธีการนี้จึงเป็นการเข้ารหัสตัวเลข เมื่อจะนำไปใช้ก็จะมีการถอดรหัสโดยนำแมนติส-สามาคูณกับเอ็กซ์โปเนนต์จะเห็นได้ว่า $2^3 \times 1.25 = 10$

อย่างไรก็ตามในเอ็กซโปเนนต์ก็ไม่ได้เก็บเลข 3 นี้ลงไปตรง ๆ เนื่องจากส่วนของเอ็กซโปเนนต์นี้ไม่มีบิตใดที่ใช้บอกเครื่องหมายเลย ดังนั้น การเก็บเลขเอ็กซโปเนนต์จะใช้วิธีบวกเลขค่าหนึ่งซึ่งเรียกว่าไบแอส 8087 ใช้เลข 3FFFH เป็นเลขไบแอส เมื่อนำมาบวกกับ 3 จึงมีค่าเป็น 4002H ดังนั้นเลข 10 จะถูกเก็บไว้ดังรูปที่ 3

ฐานสิบ : 10
ฐานสิบหก : A
จำนวนเต็มฐานสอง 64 บิต : 0000000000000000...0000000000001010
จำนวนจริง 80 บิตแบบ 8087 : 1010000000000000...0000000000000000
แสดงเฉพาะซิกนิฟิแคนด์

รูปที่ 2 การเก็บเลข 10 ในวิธีต่าง ๆ



รูปที่ 3 การแทนเลข 10 ในข้อมูลขนาด 80 บิต ของ 8087

ตารางที่ 1 ขนาดและความละเอียดของข้อมูลแบบต่างๆ

ชนิดข้อมูล	จำนวนบิต	ช่วงตัวเลขโดยประมาณ
จำนวนจริงชั่วคราว	80	$3.4\times10^{-4932} \le N \le 1.2 \times 10^{4932}$
แพคบีซีดี	80	$-10^{18}-1 \le N \le 10^{18}-1$
จำนวนจริงแบบยาว	64	$4.19\times10^{-307} \le N \le 1.67 \times 10^{308}$
จำนวนเต็มแบบยาว	64	$-9\times10^{18} \le N \le +9 \times 10^{18}$
จำนวนจริงแบบสั้น	32	$8.43\times10^{-37} \le N \le 3.37 \times 10^{38}$
จำนวนเต็มแบบสั้น	32	$-2\times10^{9} \le N \le +2 \times 10^{9}$
จำนวนเต็มแบบเวิร์ด	16	$-32{,}768 \le N \le +32{,}767$

ข้อแนะนำในการใช้ข้อมูลแบบต่าง ๆ

ในตารางที่ 1 จะสรุปลักษณะของข้อมูลทั้ง 7 ชนิด และช่วงของตัวเลขโดยประมาณที่เก็บได้ การเลือกใช้ชนิดของข้อมูลนั้นขึ้นกับลักษณะของข้อมูลเองว่าเป็นจำนวนเต็มหรือจำนวนจริง ในกรณีที่เป็นจำนวนจริงก็ให้คำนึงถึงความละเอียดของตัวเลขที่ต้องการเก็บและขนาดของตัวเลขว่าใหญ่เพียงใด อย่าลืมว่าการใช้ข้อมูลที่มีจำนวนบิตมากเกินความจำเป็นย่อมทำให้โปรแกรมเสียเวลาการทำงานเพิ่มขึ้นและเปลืองเนื้อที่หน่วยความจำมากขึ้นด้วย

ชุดคำสั่งของ 8087

คำสั่งของ 8087 เป็นเสมือนกับคำสั่งส่วนขยายของซีพียู เราจะสังเกตและจำคำสั่งใน 8087 ได้ง่ายเพราะทุกคำสั่งขึ้นต้นด้วยตัว F เสมอ คำสั่งทั้ง 69 คำสั่งของ 8087 จัดได้เป็นกลุ่มตามตารางที่ 2

มองลักษณะคำสั่งของ 8087

คำสั่ง 8087 ส่วนใหญ่แล้วจะเกี่ยวข้องกับรีจิสเตอร์สแต็กเสมอ โดยเฉพาะ

ตารางที่ 2 สรุปชุดคำสั่งของ 8087

หมายเหตุ	รหัสนีโมนิก	โหมดการอ้างแอดเดรส	การทำงานของคำสั่ง
	คำสั่งโอนย้ายข้อมูล(9)		
	FXCH	//d	แลกค่าในรีจิสเตอร์
	FLD	s	โหลดจำนวนจริง
(P)	FST	d	เก็บจำนวนจริง
	FILD	s	โหลดจำนวนเต็ม
(P)	FIST	d	เก็บจำนวนเต็ม
	FBLD	s	โหลดแพคบีซีดี
	FBSTP	d	เก็บแพคบีซีดี
	คำสั่งโหลดค่าคงที่(7)		
	FLDZ		โหลดค่า +0.0
	FLD1		โหลดค่า +1.0
	FLDPI		โหลดค่า Pi
	FLDL2T		โหลดค่า $\log_2 10$
	FLDL2E		โหลดค่า $\log_2 e$
	FLDLG2		โหลดค่า $\log_{10} 2$
	FLDLN2		โหลดค่า $\log_e 2$
	คำสั่งทรานเซนเดนทัล (5)		
	FPTAN		พาร์เชียลแทนเจนต์
	FPATAN		พาร์เชียลอาร์กแทนเจนต์
	F2XM1		คำนวณ $2^X - 1$
	FYL2X		คำนวณ $Y \times \log_2 X$
	FYL2XP1		คำนวณ $Y \times \log_2 (X+1)$
	คำสั่งเปรียบเทียบ(7)		
(P)	FCOM	//s	เปรียบเทียบจำนวนจริง
(P)	FICOM		เปรียบเทียบจำนวนเต็ม
(P)	FCOMPP		เปรียบเทียบและ POP สองครั้ง
(P)	FTST		ตรวจสอบสแต็กบนสุด
(P)	FXAM		ทดสอบสแต็กบนสุด
	คำสั่งคณิตศาสตร์(25)		
(P)	FADD	*	บวกจำนวนจริง
	FIADD	s	บวกจำนวนเต็ม
(P)	FSUB	*	ลบจำนวนจริง
	FISUB	s	ลบจำนวนเต็ม
(P)	FSUBR	*	ลบจำนวนจริง(สลับตัวตั้งกับตัวลบ) ▶

ตารางที่ 2 (ต่อ)

หมายเหตุ	รหัสนีโมนิก	โหมดการอ้างแอดเดรส	การทำงานของคำสั่ง
		คำสั่งคณิตศาสตร์(ต่อ)	
	FISUBR	s	ลบจำนวนเต็ม(สลับตัวตั้งกับตัวลบ)
(P)	FMUL	*	คูณจำนวนจริง
	FIMUL	s	บวกจำนวนเต็ม
(P)	FDIV	*	หารจำนวนจริง
	FIDIV	s	หารจำนวนเต็ม
(P)	FDIVR	*	หารจำนวนจริง(สลับตัวตั้งกับตัวหาร)
	FIDIVR	s	หารจำนวนเต็ม(สลับตัวตั้งกับตัวหาร)
	FSQRT		ถอดรากที่สอง
	FSCALE		สเกลค่าทีละ 2^n
	FPREM		ทำพาร์เชียลมอดูโล
	FRNDINT		ปัดเป็นจำนวนเต็ม
	FXTRACT		แยกส่วนเอ็กซโปเนนต์และซิกนิฟิแคนด์
	FABS		หาค่าสัมบูรณ์
	FCHS		เปลี่ยนเครื่องหมายบวกลบ
		คำสั่งควบคุม(16)	
(N)	FINIT		เริ่มต้นโปรเซสเซอร์
	FLDCW	s	โหลดคอนโทรลเวิร์ด
(N)	FSTCW	d	เก็บคอนโทรลเวิร์ด
(N)	FSTSW	d	เก็บสเตตัสเวิร์ด
(N)	FSTENV	d	เก็บเอนวิโรนเมนต์
	FLDENV	s	โหลดเอนริโรนเมนต์
(N)	FSAVE	d	เก็บสถานะ
	FRSTOR	s	เรียกคืนสถานะ
	FINCSTP		เพิ่มสแต็กพอยน์เตอร์
	FDECSTP		ลดสแต็กพอยน์เตอร์
	FFREE		ล้างค่ารีจิสเตอร์
	FNOP	d	ไม่มีการทำงาน
	FWAIT		ให้ซีพียูรอ
(N)	FDISI		ดิสเอเบิ้ล อินเตอร์รัพต์
(N)	FENI		อีนาเบิ้ล อินเตอร์รัพต์
(N)	FCLEX		ยกเลิกเอ็กเซปชัน

*รูปคำสั่ง FADD, FSUB, FSUBR, FMUL, FDIV, FDIVR

:F<OP> หมายถึง P<OP>P ST(1) ST

:F<OP>s หมายถึง F<OP>ST, <memory>

:F<OP>d, s หมายถึง ใช้เฉพาะ d,s เท่านั้น

:F<OP>P d,s หมายถึง ใช้เฉพาะ d,s เท่านั้น

(P) หมายถึง คำสั่งอาจอยู่ในรูป F<OP> หรือ F<OP>P ก็ได้

(N) หมายถึง คำสั่งอาจอยู่ในรูป F<OP> หรือ FN<OP> ก็ได้

s หมายถึง ค่าต้นทาง

d หมายถึง ค่าปลายทาง

//s จะมีค่าต้นทางหรือไม่มีก็ได้

//d จะมีค่าปลายทางหรือไม่มีก็ได้

คำสั่งอ่านข้อมูลและเก็บข้อมูล ซึ่งมักใช้สแต็กเป็นโอเปอแรนด์ บางคำสั่งก็อาจจะใช้เพียงสแต็กบนสุด เป็นโอเปอแรนด์ เช่น คำสั่ง FSQRT และ FABS บางคำสั่งก็ใช้ทั้งสแต็กบนสุดและสแต็กถัดไป เช่น FSCALE และ F2XM1

ถ้าเราสังเกตคำสั่งเกี่ยวกับการโหลดข้อมูลจะเห็นว่าเป็นการ push ข้อมูลเข้าสู่สแต็กนั่นเอง การโหลดข้อมูลจะเป็นแบบจำนวนจริงหรือจำนวนเต็มหรือแบบ BCD ใช้สังเกตได้ที่ตัวคำสั่งเอง เช่น

FLD เป็นการโหลดจำนวนจริง
FILD หมายถึง โหลดจำนวนเต็ม
FBLD หมายถึง โหลดตัวเลขบีซีดี

ส่วนคำสั่งเก็บข้อมูลก็เช่นเดียวกัน คำสั่งเก็บข้อมูลมีข้อพิเศษก็คือ บางคำสั่งหลังจากที่เก็บข้อมูลแล้วจะ pop ข้อมูลบนสุดทิ้งไป คำสั่งประเภทนี้สังเกตได้จากตัว P ที่อยู่ต่อท้ายคำสั่ง

เราลองตีความหมายคำสั่งบางคำสั่งต่อไปนี้

- **FLD SHORT X**

หมายถึงโหลดจำนวนจริงแบบ short จากหน่วยความจำตำแหน่ง X ไปเก็บที่สแต็กบนสุด

- **FLD ST(4)**

หมายถึง โหลดค่าในรีจิสเตอร์สแต็ก ST(4) ไปไว้ที่สแต็กตัวบนสุด (ST (0))

- **FADD ST (2), ST**

หมายถึงนำค่าในรีจิสเตอร์สแต็ก ST(2) บวกกับ ST(0) แล้ว นำไปเก็บแทนที่ ST(2)

- **FADDP ST(1), ST(0)**

หมายถึงนำค่าในรีจิสเตอร์ ST(1) บวกกับ ST(0) แล้วเก็บไว้ที่ ST(1) แล้ว POP ค่า ST(0) ทิ้งไป ผลลัพธ์ของการบวกซึ่งเดิมอยู่ที่ ST(1) จะเลื่อนไปอยู่ที่ ST(0)

คราวนี้เพื่อให้เราทำความเข้าใจกับคำสั่งของ 8087 ดีขึ้น ให้เรามาดูตัวอย่างง่าย ๆ โดยสมมติว่าเรามีตัวแปร 3 ตัว คือ X,Y และ Z ซึ่งกำหนดค่าไว้แล้ว (ความจริง X,Y และ Z คือ แอดเดรสของหน่วยความจำนั่นเอง) และสมมติให้ X เป็นจำนวนจริงแบบสั้น Y เป็นจำนวนเต็มแบบเวิร์ด และ Z เป็นจำนวนจริงแบบยาว และเราจะคำนวณค่า

$$X = \frac{(X^2+Y^2)}{X+Y} + \sqrt{(Z-Y)}$$

เราก็เขียนคำสั่งได้ง่าย ๆ ดังรูปที่ 4

```
FLD LONG Z      ; โหลด Z สู่สแต็กบนสุด
FISUB WORD Y    ; นำค่า Y ไปลบจาก Z แล้วเก็บผลไว้ที่สแต็กบนสุด
FSQRT           ; ถอดรากของ (Z-Y)
FLD SHORT X     ; โหลด X
FMUL ST, ST(0)  ; คูณกับตัวเองได้ X^2
FILD WORD Y     ; โหลด Y
FMUL ST, ST(0)  ; คูณกับตัวเองได้ Y^2
FADD            ; คำนวณ X^2 + Y^2
FLD SHORT X     ; โหลดค่า X ใหม่อีกครั้ง
FIADD WORD Y;   บวก X กับ Y
FDIV            ; นำ (X^2+Y^2)/X+Y เก็บไว้สแต็กบนสุด
FADD            ; นำผลมาบวกกับ √(Z-Y)
FST SHORT X     ; เก็บผลที่ได้ไว้ที่ X
```

รูปที่ 4 ตัวอย่างคำสั่ง 8087 คำนวณสูตร

คำสั่งและความสัมพันธ์กับรีจิสเตอร์สแต็ก

จากโปรแกรมในรูปที่ 4 บางครั้งเราอาจเกิดความสับสนขึ้นได้ว่าค่าในสแต็กมีการเปลี่ยนแปลงไปอย่างไรบ้าง ซึ่งจุดนี้เป็นจุดสำคัญที่ต้องทำความเข้าใจให้ดีเพราะเป็นพื้นฐานสำคัญในการเขียนโปรแกรม ขอให้เราพักความสนใจโปรแกรมในรูปที่ 4 ไว้และหันมาดูโปรแกรมสั้น ๆ 4 คำสั่งต่อไปนี้

```
FLD<mem-1>  ; โหลดค่าจากเมมโมรี- 1
FLD<mem-2>  ; โหลดค่าจากเมมโมรี- 2
FADD        ; หมายถึง FADDP ST(1), ST
FSTP<mem-3> ; เก็บผลลัพธ์ไปไว้ที่เมมโมรี- 3
```

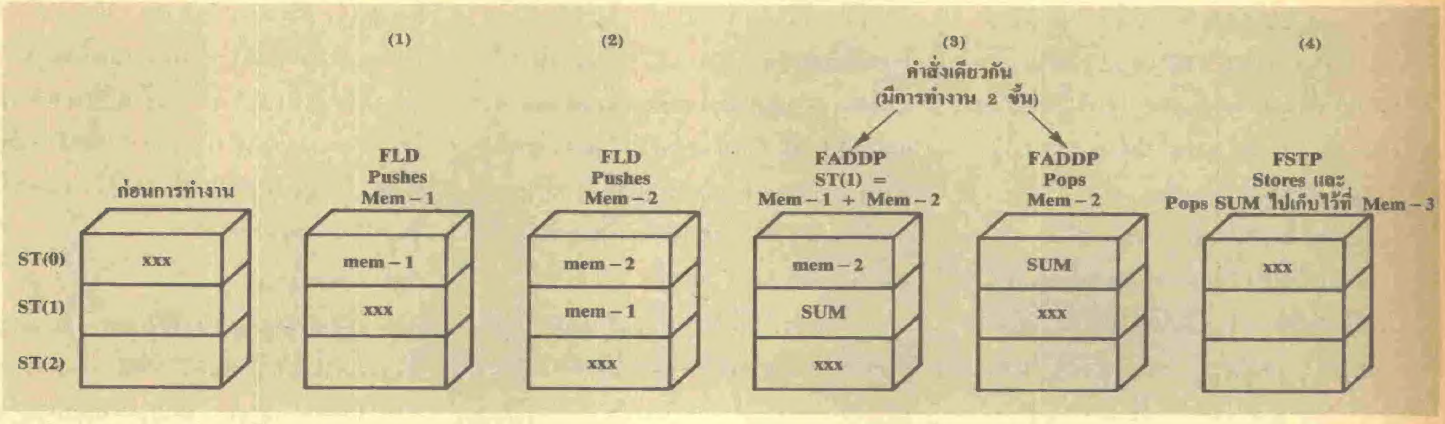
ผลของคำสั่งทั้ง 4 นี้จะให้ผลต่อสแต็กดังในรูปที่ 5 ขอให้สังเกตว่าการโหลดหรือนัยหนึ่งคือ การ push นั้นจะทำให้เกิดการเลื่อนข้อมูลในสแต็กเสมอ เช่น เริ่มต้น ค่า ST(0) = XXX เมื่อ 8087 ทำคำสั่ง FLD<mem-1> ทำให้ ST(0) มีค่า<mem-2>โดยค่า XXX ถูกเลื่อนลงไปอยู่ที่ ST(1)

ดังนั้น ถ้าย้อนกลับไปพิจารณาโปรแกรมในรูปที่ 4 จะพบว่าค่าในรีจิสเตอร์สแต็กจะแปรเปลี่ยนไปตามคำสั่งซึ่งแสดงได้ดังรูปที่ 6

โดยสรุปแล้วสิ่งที่สำคัญที่เราต้องคำนึงถึงในการใช้รีจิสเตอร์สแต็กก็คือ คำสั่งนั้นทิ้งผลลัพธ์ไว้ที่สแต็กอย่างไร นอกจากนี้ ก็ต้องระมัดระวังสแต็กโอเวอร์โฟลว์เนื่องจากมีให้ใช้เพียง 8 ตัวเท่านั้น อย่างไรก็ตามถ้าเกิดเหตุการณ์นี้ขึ้น 8087 จะบอกสถานะให้เรารู้ทางสเตตัสเวิร์ด

คำสั่งคณิตศาสตร์และฟังก์ชันทรานเซนเดนทัล

คำสั่งของ 8087 เกี่ยวกับการคำนวณทางคณิตศาสตร์และฟังก์ชัน



รูปที่ 5 ค่าต่างๆที่เกิดขึ้นในรีจิสเตอร์สแต็ก โดยสมมติให้ก่อนการทำงานมีค่า xxx ใน ST(0)

	FLD	FISUB	FSQRT	FLD	FMUL	FILD
ST0	Z	Z−Y	$\sqrt{Z-Y}$	X	X^2	Y
ST1				$\sqrt{Z-Y}$	$\sqrt{Z-Y}$	X^2
ST2						$\sqrt{Z-Y}$

	FMUL	FADD	FLD	FIADD	FDIV	FADD
ST0	Y^2	X^2+Y^2	X	X+Y	$X^2+Y^2/X-Y$	$X^2+Y^2/X-Y+\sqrt{Z-Y}$
ST1	X^2	$\sqrt{Z-Y}$	X^2+Y^2	X^2+Y^2	$\sqrt{Z-Y}$	
ST2	$\sqrt{Z-Y}$		$\sqrt{Z-Y}$	$\sqrt{Z-Y}$		

รูปที่ 6 ค่าในรีจิสเตอร์สแต็ก เมื่อทำงานตามโปรแกรมในรูปที่ 4

ทรานเซนเดนทัลเป็นกลุ่มคำสั่งที่มีความสำคัญที่สุดกลุ่มหนึ่ง โดยทั่วไปแล้วคำสั่งกลุ่มนี้จะกระทำต่อสแต็กบนสุด และก็มีบางคำสั่งซึ่งมีผลต่อ ST1 ด้วย คำสั่งในกลุ่มนี้จะขอแยกกล่าวดังนี้

คำสั่งทางคณิตศาสตร์

● **FSQRT ;ST ← Square Root (ST)**

เป็นคำสั่งหารากที่สองของรีจิสเตอร์ ST

● **FSCALE ;ST←ST*2^{ST1}**

สเกลค่า ST ด้วย 2^{ST1}

● **FPREM ; ST ← ST Partial mod (ST1)**

เป็นการหาเลขมอดูโลของ ST แต่เป็นแบบพาร์เชียล คือ จะหาเลขมอดูโลได้เฉพาะตัวเลขที่ไม่มากนัก เนื่องจากการทำคำสั่งแต่ละครั้งจะลดค่าตัวเลขได้ไม่เกิน 2^{64} ดังนั้นแล้ว ถ้า ST มีค่าเกิน 2^{64} ผลลัพธ์ของการมอดูโลหรือหาเศษจะยังไม่ถูกต้อง จะต้องทำพาร์เชียลต่อไปใหม่ คำสั่ง FPREM จะเซตแฟล็ก C2 ให้มีค่าเป็น "1" ถ้าการทำพาร์เชียลยังไม่สมบูรณ์ คำสั่งนี้อาจใช้ประโยชน์ในการกำหนดขอบเขตของตัวเลขให้อยู่ในช่วงที่จะนำไปใช้งานได้ เช่น จำกัดค่าของมุมที่จะนำไปใช้ในคำสั่งทางตรีโกณมิติ

● **FRNDINT ;ST ← Integer (ST)**

เป็นคำสั่งปัดเลขให้เป็นจำนวนเต็ม

● **FXTRACT ; ST ← Fraction of ST**

ST1 ← Exponent of ST

ใช้ในการแยกส่วนของเอ็กซโปเนนต์และซิกนิฟิแคนด์ไปเก็บไว้ที่ ST1 และ ST ตามลำดับ

● **FABS ; ST ← Absolute Value of ST**

เป็นคำสั่งหาค่าสัมบูรณ์ของ ST และผลลัพธ์ที่ได้จะเก็บไว้ที่ ST

● **FXTRACT ; ST ← Fraction of ST**

● **FCHS ; ST ← −ST**

เป็นคำสั่งกลับเครื่องหมายของตัวเลขใน ST

คำสั่งฟังก์ชันทรานเซนเดนทัล

● **FPTAN ;ST←X**

ST1←Y

เมื่อ Y/X = TAN (ANGLE) คำสั่งนี้ เป็นคำสั่งหาค่าพาร์เชียลแทนเจนต์ของมุมและเก็บค่าที่ได้ไว้ที่สแต็กบนสุด ที่เรียกว่าพาร์เชียลนี้เนื่องจากมุมที่จะคำนวณต้องอยู่ในช่วง

0< ANGLE < PI/4

● **FPATAN ;ST←ARCTAN (Y/X) = ARCTAN (ST1/ST)**
คำสั่งนี้เป็นส่วนกลับของ FPTAN โดยที่ค่า X, Y ต้องเป็นไปตามอสมการ

$0 < Y < X < \infty$

● **FZXM1 ;ST←$2^{ST} - 1$**
คำสั่งนี้เป็นคำสั่งทางเอ็กซโปเนนเชียล โดยใช้ค่าใน ST เป็นเลขชี้กำลัง อย่างไรก็ตามค่า ST นี้ จะต้องอยู่ในช่วง $0 \leqslant ST \leqslant 0.5$ ในกรณีที่ ST มากกว่า 0.5 จะต้องใช้คำสั่ง FSCALE เข้ามาช่วย

● **FYL2X ;ST←$Y*LOG_2X = ST*LOG_2ST$**
คำสั่งนี้จะหาค่าลอการิทึมฐานสองของ ST แล้วนำมาคูณกับ ST1 ค่า X และ Y จะต้องเป็นไปตาม

$0 < X < \infty$

$-\infty < Y < \infty$

● **FYL2XP1 ;ST←$Y.LOG_2 (X+1) = ST1.LOG_2(ST+1)$**
คำสั่งนี้ มีลักษณะเช่นเดียวกับ FYL2X ต่างกันแต่เพียง ST จะถูกนำมาบวกด้วย 1 ก่อนที่จะหาค่าลอการิทึมฐานสอง

ความสำคัญของคำสั่ง FWAIT

ลักษณะเด่นที่สำคัญประการหนึ่งของ 8087 ก็คือ สามารถทำงานตามคำสั่งของตัวเองขนานไปกับการทำงานของซีพียูได้ และซีพียูก็ยังสามารถทำงานของตัวเองได้เช่นเดียวกัน แต่ปัญหาที่เราต้องคำนึงถึงก็คือในกรณีที่ 8087 กำลังทำคำสั่งใดๆอยู่และก็เป็นจังหวะเดียวกับที่ซีพียูเฟตช์คำสั่งใหม่เข้ามาและพบว่าเป็นคำสั่ง 8087 อีก ถ้าซีพียูส่งการทำงานไปให้ 8087 ในขณะที่ 8087 ยังไม่พร้อมที่จะทำคำสั่งต่อไป ผลย่อมทำให้เกิดการผิดพลาดขึ้น

โดยปกติแล้ว ซีพียูจะต้องรอให้ 8087 จบการทำงานคำสั่งใดๆก่อนที่จะเอ็กซีคิวต์คำสั่งต่อไปได้ วิธีการนี้ทำได้โดยเพิ่มคำสั่ง FWAIT ไว้ก่อนหน้าคำสั่งใด ๆ ดังนั้นเมื่อซีพียูเอ็กซีคิวต์คำสั่ง FWAIT นี้ ซีพียูจะรอให้ 8087 เสร็จสิ้นการทำงานคำสั่งเสียก่อน (โดยมองผ่านขา TEST ของซีพียูที่ต่อเชื่อมกับขา $\overline{BUSY}$ ของ 8087)

อย่างไรก็ตามเมื่อเราเขียนโปรแกรม 8087 และใช้แมคโครแอสเซมเบลอร์ MASM ตัว MASM จะสร้างคำสั่งFWAIT นำหน้ารหัสคำสั่ง 8087 ทุกคำสั่ง ยกเว้นบางคำสั่งที่ MASM ทราบโดยตัวเองว่าไม่จำเป็นต้องมีการใช้ FWAIT จะทำให้ไม่มีการสร้าง FWAIT นำหน้าไว้ คำสั่งประเภทนี้ได้แก่ คำสั่งที่มีอักษร (N) ตามตารางที่ 2

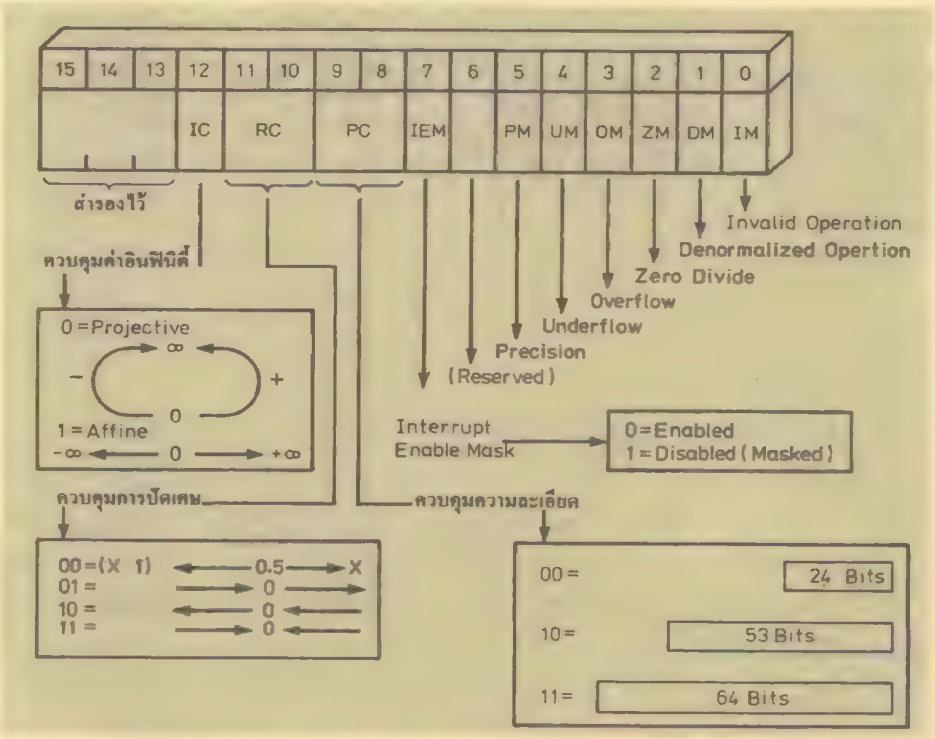
จะควบคุม 8087 ได้อย่างไร

นอกจากรีจิสเตอร์สแต็ก 8 ตัวแล้ว 8087 ยังมีรีจิสเตอร์อีก 4 ตัวที่ออกแบบไว้ให้ผู้ใช้ติดต่อได้ รีจิสเตอร์นี้ได้แก่ คอนโทรลเวิร์ด สเตตัสเวิร์ด อินสตรัคชันพอยน์เตอร์ และดาต้าพอยน์เตอร์ 8087 ยังมีรีจิสเตอร์อีกตัวหนึ่งคือ แท็กเวิร์ด แต่เป็นรีจิสเตอร์ที่ใช้เฉพาะภายในเท่านั้น ในหัวข้อต่อไปนี้เราจะได้พูดถึงการใช้งานรีจิสเตอร์นี้ในการควบคุม 8087

คอนโทรลเวิร์ด

เป็นรีจิสเตอร์ซึ่งใช้กำหนดลักษณะการทำงานของ 8087 เช่น การตรวจสอบข้อผิดพลาด การควบคุมความละเอียดของตัวเลข หรือผลของการคำนวณเลข เป็นต้น

คอนโทรลเวิร์ดขนาด 16 บิตนี้ จะแบ่งแต่ละบิตเพื่อควบคุมการทำงานดังรูปที่ 7 จะเห็นได้ชัดว่า 8 บิตล่าง (บิต 0 ถึง 7) เป็นบิตที่ใช้ควบคุมการทำงานเมื่อมีการผิดพลาด ซึ่งมีได้หลายสาเหตุ เป็นต้นว่าถ้ามีสิ่งผิดปกติเกิดขึ้นจากการคำนวณเช่นหารด้วยศูนย์(Zero Divide) 8087 จะตรวจดูว่าบิต ZM (บิต 2) แอคทีฟหรือมีค่า "1" หรือไม่ ถ้าเป็น "1" ก็จะมีการสร้างสัญญาณ



รูปที่ 7 บิตต่างๆ ในรีจิสเตอร์ควบคุม

อินเตอร์รัพต์ส่งไปบอกซีพียูทันที และสำหรับบิตอื่นก็ควบคุมการผิดพลาดด้วยวิธีการเดียวกัน อย่างไรก็ตามเราอาจดิสเอเบิ้ลหรือระงับสัญญาณอินเตอร์รัพต์นี้ได้โดยไม่สนใจว่าจะมีข้อผิดพลาดใดเกิดขึ้น ซึ่งทำได้โดยให้บิตที่ 7 มีค่าเป็น "1" หรือดิสเอเบิ้ลอินเตอร์รัพต์

สำหรับบิต 8-9 ใช้ควบคุมความละเอียดของตัวเลขว่ามีได้ถึงกี่บิต ส่วนบิต 10-11 ควบคุมการปัดเศษว่าจะให้ปัดเข้าใกล้จำนวนเต็ม(00) หรือปัดขึ้น(01) ปัดลง(10) หรือปัดทิ้ง(11) และบิต 12 ใช้ควบคุมตัวเลขเมื่อการคำนวณมีค่ามากถึงอินฟินิตี้ ถ้าบิตนี้เป็น "0" หมายถึงไม่สนใจเครื่องหมายว่าเป็นอินฟินิตี้บวกหรือลบ 8087 จะถือว่าเป็นอินฟินิตี้ที่ไม่มีเครื่องหมาย(อีกนัยหนึ่งคืออินฟินิตี้บวกนั่นเอง) แต่ถ้าบิตนี้เป็น "1" หมายถึงค่าอินฟินิตี้นี้เป็นค่าที่มีเครื่องหมายบวกลบ

การกำหนดค่าให้รีจิสเตอร์ควบคุมนี้ทำได้โดยใช้คำสั่ง FLDCW ดังตัวอย่างโปรแกรม

```
CW87 DW 03BFH;8087 control
                word
     •
     •
     •
     FLDCW CW87; Load 8087
                Control word
```

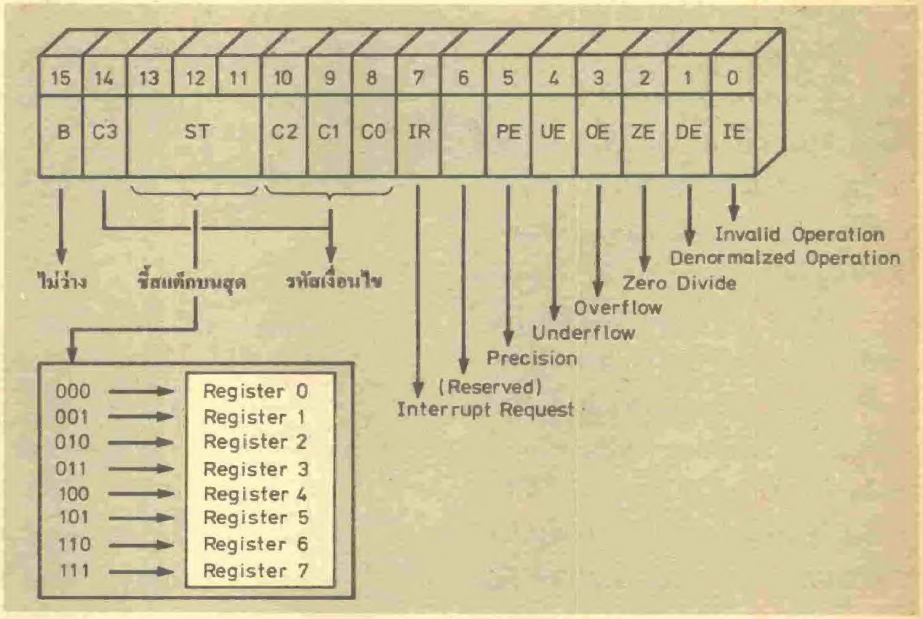
ค่า 03BF นี้ โดยปกติเป็นค่าที่ 8087 กำหนดเริ่มต้นให้

สเตตัสเวิร์ด

รีจิสเตอร์ตัวนี้ใช้แสดงผลของการทำงาน 4 อย่างคือ

1. บ่งบอกสถานะไม่ว่าง
2. ชี้สแต็กบนสุด
3. แฟล็กเนื่องจากผลของคำสั่ง FCOM, FTST และ FXAM
4. บ่งบอกสถานะผิดพลาด

สำหรับเงื่อนไขที่ทำให้เกิดแฟล็กต่างๆ จะแสดงไว้ในตารางที่ 3



รูปที่ 8 บิตต่างๆ ในรีจิสเตอร์สถานะ

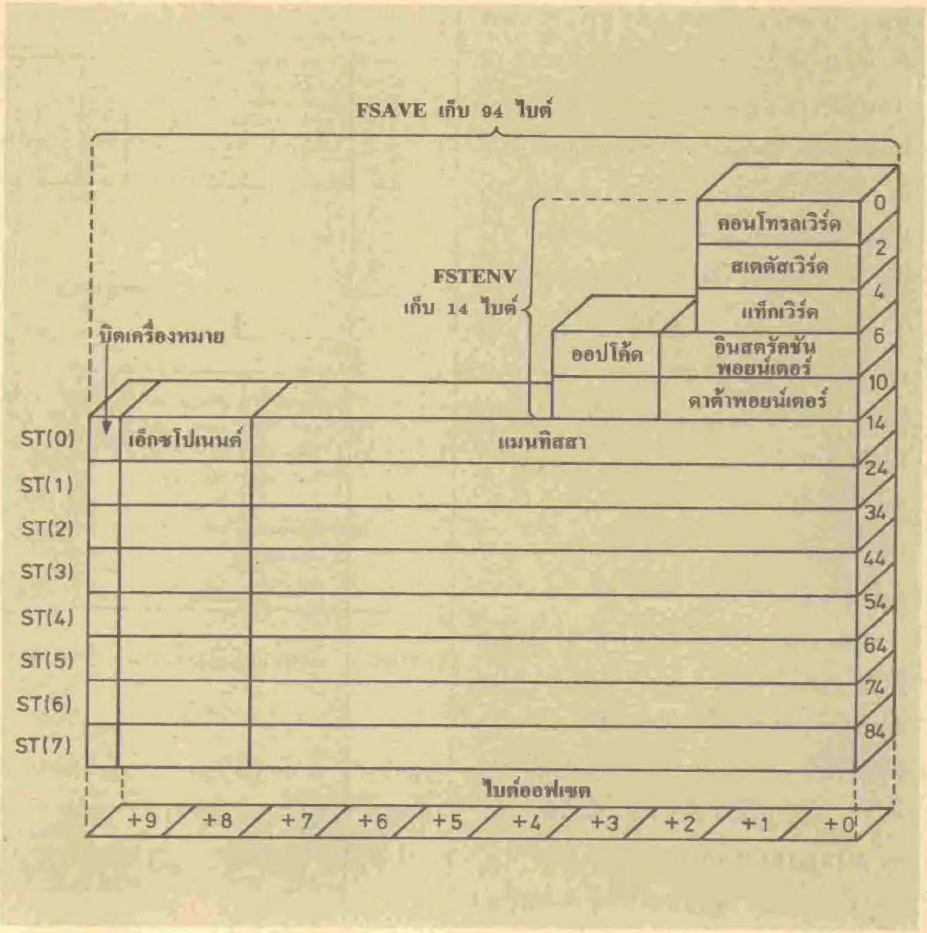
ตารางที่ 3 เงื่อนไขและผลที่เกิดต่อแฟล็ก

	รหัสเงื่อนไข				ผลลัพธ์
	C3	C2	C1	C0	
F	0	0	D	0	ST > source
C	0	0	O	1	ST < source
O	1	0	N	0	ST = source
M	1	1	'T	1	ST ? source
F	0	0	C	0	ST > 0.0
T	0	0	A	1	ST < 0.0
S	1	0	R	0	ST = 0.0
T	1	1	E	1	ST ? 0.0
F	0	0	0	0	+Unnormal
X	0	0	0	1	+NAN
A	0	0	1	0	−Unnormal
M	0	0	1	1	−NAN
	0	1	0	0	+Normal
	0	1	0	1	+Infinity
	0	1	1	1	−Infinity
	1	0	0	0	+Zero
	1	0	0	1	Empty
	1	0	1	0	−Zero
	1	0	1	1	Empty
	1	1	0	0	+Denormal
	1	1	0	1	Empty
	1	1	1	0	−Denormal
	1	1	1	1	Empty

NAN : Not A Number เกิดขึ้นเมื่อการทำงานของคำสั่งไม่นิยาม เช่นหารากที่สองของเลขลบ เป็นต้น

อินสตรัคชันและดาต้าพอยน์เตอร์

รีจิสเตอร์สุดท้ายสองตัวนี้จะใช้ในกรณีที่เราควบคุมการอินเตอร์รัพต์แล้วให้กระโดดไปยังรูทีนอินเตอร์รัพต์ของเราเอง รูทีนนี้จะต้องอ่านค่ารีจิสเตอร์สถานะเข้ามาตรวจดูว่าสาเหตุของปัญหาคืออะไร และปัญหาเกิดขึ้นที่คำสั่งใด คำสั่งนี้มีการใช้ข้อมูลที่ไหน อินสตรัคชันและดาต้าพอยน์เตอร์นี้ก็จะเป็นตัวชี้ว่าแอดเดรสของคำสั่งและข้อมูลนี้อยู่ที่ใด วิธีการที่เราจะติดต่อกับอินสตรัคชันและดาต้าพอยน์เตอร์นี้ก็ต้องติดต่อผ่านหน่วยความจำโดยใช้คำสั่ง FSTENV หรือ FSAVE ซึ่งจะเก็บสถานะต่าง ๆ จาก 8087 ไปสู่หน่วยความจำดังรูปที่ 9



รูปที่ 9
ผลของคำสั่ง FSTENV FSAVE ทำให้มีการเก็บค่าต่างๆ ลงสู่หน่วยความจำ ▶

ซอฟต์แวร์และการเขียนโปรแกรม

ในภาคนี้เราจะได้กล่าวถึงการพัฒนาโปรแกรม 8087 ภายใต้เอ็มเอสดอสโดยใช้แมคโครแอสเซมเบลอร์ MASM รวมทั้งตัวอย่างโปรแกรมการแก้ปัญหาทางคณิตศาสตร์

การใช้ MASM กับ 8087

การเขียนโปรแกรม 8087 ทำได้เช่นเดียวกับการเขียนโปรแกรม 8086 เราสามารถใช้ MASM ในการแอสเซมเบลอร์นีโมนิกของ 8087 ได้ แต่จำเป็นต้องใช้พารามิเตอร์/r ในการแอสเซมเบลอร์ไฟล์ที่มีรหัสนีโมนิกของ 8087 ด้วย ทั้งนี้เนื่องจากในโหมดปกติแล้ว MASM ไม่สามารถตีความหมายนีโมนิก 8087 ได้ เราจึงต้องใช้พารามิเตอร์นี้ บอกว่าไฟล์ของเรามีรหัส 8087 อยู่ด้วย ตัวอย่างเช่นการแอสเซมไฟล์ EX8087.ASM ให้ใช้คำสั่ง

A>MASM EX8087/r

แบบข้อมูล 8087 กับ MASM

เราทราบมาแล้วว่า 8087 ใช้งานได้กับตัวเลขต่างๆได้ 8 ชนิด ในการเรียกใช้ตัวเลขต่างๆ เหล่านี้ทำได้โดย

กำหนดแบบข้อมูลใน MASM ให้ตรงกับแบบตัวเลขนั้น ในตารางที่ 1 แสดงให้เห็นถึงแบบข้อมูลใน 8087 ซึ่งตรงกับแบบข้อมูลใน MASM

อย่างไรก็ตามมีข้อพึงระวังในการใช้เลขจำนวนจริงทั้งแบบสั้นและยาว ซึ่ง MASM จะใช้แตกต่างจาก 8087 ในรูปที่ 1 จะบอกถึงลักษณะที่ MASM เก็บตัวเลข ซึ่งเห็นได้ชัดว่าแตกต่างจากจำนวนจริงแบบสั้น และแบบยาว ตามรูปที่ 1 ในตอนที่แล้ว ดังนั้นในกรณีนี้ผู้เขียนโปรแกรมจำเป็นต้องเขียนโปรแกรมย่อยแปลงรูปแบบข้อมูลเองให้ถูกต้อง

การคำนวณค่าคณิตศาสตร์

ชุดคำสั่งที่นับเป็นหัวใจสำคัญของ 8087 ได้แก่ ชุดคำสั่งทางคณิตศาสตร์ ซึ่งมีคำสั่งโหลดค่าคงที่และคำนวณค่าต่างๆ เช่น ค่าคงที่ $\log_2 10$, $\log_2 e$ หรือ $\log_{10} 2$ เป็นต้น อย่างไรก็ตามสมการทางคณิตศาสตร์ที่ต้องคำนวณอาจซับซ้อน หรือฟังก์ชันทางคณิตศาสตร์ที่ไม่มีอยู่ในชุดคำสั่ง แต่เราก็อาศัยความรู้ทางคณิตศาสตร์เข้ามาแก้ไขปัญหาได้เช่น การคำนวณ 10^x, e^x, y^x, $\log_{10} x$ หรือ $\log_e x$ ซึ่งไม่มีอยู่ในชุดคำสั่งโดยตรง แต่เราคำนวณได้จาก

$$10^x = 2^{x*\log_2 10}$$

$$e^x = 2^{x*\log_2 e}$$

$$y^x = 2^{x*\log_2 y}$$

$$\log_{10} x = \log_{10} 2 * \log_2 x$$

$$\log_e x = \log_e 2 * \log_2 x$$

จะเห็นได้ว่าเราสามารถใช้ชุดคำสั่งพื้นฐานมาประกอบกันขึ้นเพื่อคำนวณค่าต่างๆได้เสมอ โปรแกรมในรูปที่ 2 เป็นตัวอย่างโปรแกรมย่อยแสดงการคำนวณ 10^x

คำสั่ง 8087 สองคำสั่งแรกจะสร้างตัวเลขชี้กำลังของ 2 ด้วยการโหลดค่า $\log_2 10$ แล้วคูณด้วยค่า X เราจะเรียกค่าที่ได้นี้ว่า E

ขั้นตอนต่อไปก็เพียงแต่นำค่า E นี้มาใช้กับคำสั่ง F2XM1 แต่ค่าที่จะนำมาใช้เป็นตัวชี้กำลัง สำหรับคำสั่ง F2XM1 จะต้องอยู่ในช่วง 0 ถึง 0.5 ดังนั้นแล้วเราจำเป็นต้องดัดแปลงคำสั่งโดยการแยกค่า E ออกเป็น 2 ส่วนคือ จำนวนเต็มและเศษ เราจะใช้คำสั่ง FSCALE สำหรับจัดการจำนวนเต็ม และ F2XM1 จัดการกับเศษ

ก่อนการแยกค่า E จะมีการปรับคอนโทรลเวิร์ดใหม่โดยปรับการปัดเศษให้เป็นแบบปัดลง ขอให้สังเกตว่าเราใช้คำสั่ง FWAIT ตามหลังคำสั่ง FNSTCW เพื่อให้แน่ใจว่า 8087 ได้เซฟค่าคอนโทรลเวิร์ดไว้เรียบร้อยแล้ว

หลังจากนั้นเราก็เริ่มแยก E ออกเป็นสองส่วนโดยแยกส่วนของจำนวนเต็มด้วยการใช้คำสั่ง FRNDINT แล้วเก็บไว้ที่ I สำหรับการหาเศษก็เพียงแต่เอาค่า E ตั้ง ลบด้วย I ก็จะได้ค่า F ซึ่งเป็นเศษนั่นเอง

ในขณะนี้เราจะได้ค่า $E = I + F$ และจะคำนวณค่า 2^E จาก

$$2^E = 2^{(I+F)} = 2^I * 2^F$$

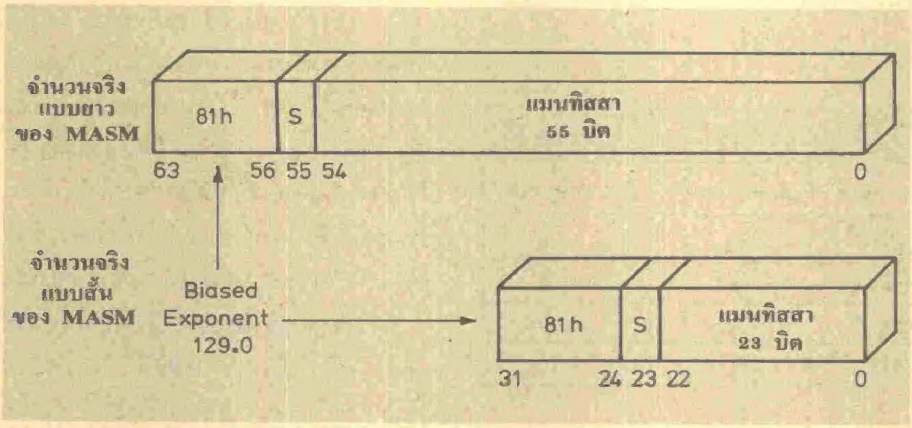
ต่อจากนั้นจึงคำนวณ 2^F และนำ 2^F มาคูณด้วย 2^I (ด้วย FSCALE) ผลลัพธ์สุดท้ายจะทิ้งอยู่ในรีจิสเตอร์ ST(0)

ข้อแนะนำการเลือกใช้ 8087/80287

ในขณะนี้ มีการผลิต 8087 ออกมาหลายเกรดหลายรุ่น รุ่นใหม่ๆก็จะ

ตารางที่ 1 แบบข้อมูลของ MASM กับ 8087 ที่สมนัยกัน

ชนิดข้อมูลของ 8087	ชนิดข้อมูลของ 8086	ขนาดไบต์	MASM ไดเรกทีฟ	ชื่อโลเปอแรนด์	ความเข้ากันได้กับ 8087
Word integer	word	2	dw	word ptr	Yes
Short integer	double word	4	dd	dword ptr	Yes
Short real	double word	4	dd	dword ptr	No
Long integer	quad word	8	dq	qword ptr	Yes
Long real	quad word	8	dq	qword ptr	No
Packed BCD	ten byte	10	dt	tbyte ptr	"R" form
Floating real	ten byte	10	dt	tbyte ptr	Yes



รูปที่ 1 ลักษณะการเก็บจำนวนจริงแบบสั้นและแบบยาวของ MASM ซึ่งต่างกับ 8087

```
code    segment  public
        assume   cs:code,ds:code
old_cw  dw       ?
new_cw  dw       ?
;-------------------------------------
; This routine takes the top element of
;  the 8087 stack and raises ten to power
; Input : ST0 is X
; Output: ST0 is 10**X
;-------------------------------------
ten_to_x proc    near
                                ;------st0-----st1-----st2----
                                ;       X       ?       ?
         fld2t                  ;  log2(10)     X       ?
         fmulp   st0,st1        ;x*log2(10)=e   ?       ?
         fnstcw  old_cw         ;
         fwait                  ;
         mov     ax,old_cw      ;
         and     ax,0c00h       ;
         or      ax,0400h       ;
         mov     new_cw,ax      ;
         fldcw   new_cw         ;
         fld1                   ;       1       e       ?
         fchs                   ;      -1       e       ?
         fld     st1            ;       e      -1       e
         frndint                ; int(e)=i     -1       e
         fldcw   old_cw         ;
         fxch    st2            ;       e      -1       i
         fsub    st0,st2        ;e-i=f         -1       i
         fscale                 ;f*2**i=f/2    -1       i
         f2xm1                  ;(2**f/2)-1    -1       i
         fsubrp  st1,st0        ;2**f/2         i       ?
         fmul    st0            ;2**f           i       ?
         fscale                 ;(2**f)*(2**i)  i       ?
         fxch    st1            ;       i      2**(i+f) ?
         fcomp                  ;2**(i+f)       ?       ?
         ret                    ;10**x
ten_to_x endp
code     ends
         end
```

รูปที่ 2 โปรแกรมคำนวณ 10^X

เทคโนโลยีการผลิตแบบเก่า HMOS 1			เทคโนโลยีการผลิตแบบใหม่ HMOS 3		
หมายเลข	ความถี่สูงสุด	อุณหภูมิสูงสุด	หมายเลข	อุณหภูมิสูงสุด	ความถี่สูงสุด
C8087-4	4.00	70	C8087	70+	5.00
C8087-6	4.77	50	C8087-2	70+	8.00
C8087-3	5.00	70	C8087-1	70+	10.00
C80287-3	5.33	70	C80287-8	70+	8.10

*nn+ : สามารถใช้งานที่อุณหภูมิสูงกว่า แต่ไม่แนะนำให้ใช้

รูปที่ 3 8087 และ 80287 รุ่นต่างๆ

มีสมรรถนะดีขึ้นเป็นลำดับ โดยปกติแล้วประมาณทุกๆ 6 เดือน จะมี 8087 รุ่นใหม่ที่มีสมรรถนะดีขึ้นในด้านความเร็วที่สูงขึ้น ความร้อนที่เกิดขึ้นในชิพน้อยลง และราคาที่ถูกลง อย่างไรก็ตาม การเลือกใช้ 8087 จำเป็นต้องคำนึงถึงคุณลักษณะของตัวไมโครคอมพิวเตอร์ที่ใช้งานอยู่ด้วย โดยจำเป็นต้องเลือกรุ่นที่มีความถี่ที่เข้ากันได้ ยกตัวอย่างเช่น ไมโครคอมพิวเตอร์ไอบีเอ็มพีซี ทำงานที่ความถี่ 4.77 MHz สามารถเลือกใช้ 8087 ได้ทุกรุ่นที่มีความถี่ตั้งแต่ 4.77 ขึ้นไป เช่น C8087-1, C8087-2, C8087-3 และ C8087-6 แต่ 8087 ที่ทำงานได้ที่ความถี่สูงมากๆ ย่อมมีราคาแพงกว่า ดังนั้นเราก็ควรเลือกใช้เฉพาะที่ทำงานที่ความถี่ที่ต้องการเท่านั้น เช่น กรณีนี้อาจจะเลือก 8087-6 ตัวอย่างอื่น เช่น ไมโครคอมพิวเตอร์ไอบีเอ็มพีซี/เอกซ์ทีที่ใช้ซีพียู 8088-2 ทำงานที่ความถี่ 8 MHz อาจเลือกใช้ C8087-1 หรือ C8087-2 ก็ได้ ในรูปที่ 3 จะแสดงถึง 8087 และ 80287 รุ่นต่างๆ

บทสรุป

8087 เป็นโปรเซสเซอร์ร่วมที่มีข้อดีเด่นในด้านความสามารถคำนวณค่าทางคณิตศาสตร์ การนำมาใช้งานจึงต้องคำนึงถึงจุดประสงค์หลักในด้านนี้ โดยเฉพาะอย่างยิ่งเมื่อนำมาใช้กับไมโครคอมพิวเตอร์ร่วมกับซอฟต์แวร์จะต้องพิจารณาว่าซอฟต์แวร์นั้นมีการคำนวณทางคณิตศาสตร์ที่จำเป็นต้องใช้ 8087 หรือไม่ และซอฟต์แวร์นั้นสนับสนุนการใช้งาน 8087 หรือไม่ มิฉะนั้นแล้วการนำ 8087 มาใช้จะไม่ก่อให้เกิดประโยชน์ใดๆ เลย